End-to-End Training of Deep Visuomotor Policies

Sergey Levine*, Chelsea Finn*, Trevor Darrell, Pieter Abbeel
Department of Electrical Engineering and Computer Sciences, UC Berkeley
{svlevine,cbfinn,trevor,pabbeel}@eecs.berkeley.edu

***Abstract*—Policy search methods based on reinforcement learning and optimal control can allow robots to automatically learn a wide range of tasks. However, practical applications of policy search tend to require the policy to be supported by hand-engineered components for perception, state estimation, and low-level control. We propose a method for learning policies that map raw, low-level observations, consisting of joint angles and camera images, directly to the torques at the robot's joints. The policies are represented as deep convolutional neural networks (CNNs) with 92,000 parameters. The high dimensionality of such policies poses a tremendous challenge for policy search. To address this challenge, we develop a sensorimotor guided policy search method that can handle high-dimensional policies and partially observed tasks. We use BADMM to decompose policy search into an optimal control phase and supervised learning phase, allowing CNN policies to be trained with standard supervised learning techniques. This method can learn a number of manipulation tasks that require close coordination between vision and control, including inserting a block into a shape sorting cube, screwing on a bottle cap, fitting the claw of a toy hammer under a nail with various grasps, and placing a coat hanger on a clothes rack.**

I. Introduction

Reinforcement learning and policy search methods hold the promise of allowing robots to acquire new behaviors through experience. They have been applied to a range of robotic tasks, including manipulation [2, 13] and locomotion [5, 7, 15, 39]. However, policies learned using such methods often rely on a number of hand-engineered components for perception and low-level control. The policy might specify a trajectory in task-space, relying on hand-designed PD controllers to execute the desired motion, and a policy for manipulating objects might rely on an existing vision system to localize these objects [29]. The vision system in particular can be complex and prone to errors, and its performance is typically not improved during policy training, nor adapted to the goal of the task.

We propose a method for learning policies that directly map raw observations, including joint angles and camera images, to motor torques. The policies are trained end-to-end using real-world experience, optimizing both the control and perception components on the same measure of task performance. This allows the policy to learn goal-driven perception, which avoids the mistakes that are most costly for task performance. Learning perception and control in a general and flexible way requires a large, expressive model. Our policies are represented with convolutional neural networks (CNNs), which have 92,000 parameters and 7 layers. Deep CNN models have been shown to achieve state of the art results on a number of supervised vision tasks [8, 16, 40], but sensorimotor deep learning remains a challenging prospect. The policies are extremely high dimensional, and the control task is partially observed, since part of the state must be inferred from images.

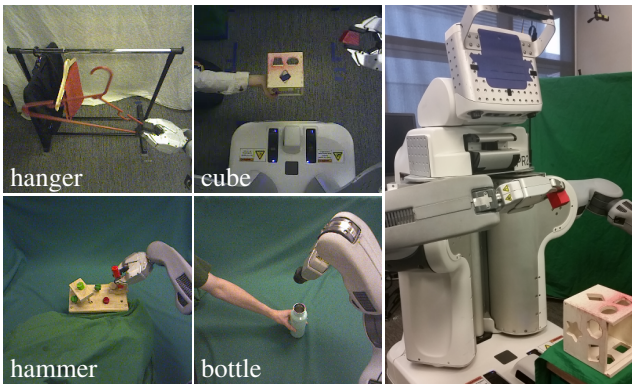


Fig. 1: Our method learns visuomotor policies that directly use camera image observations (left) to set motor torques on a PR2 robot (right).

To address these challenges, we extend the framework of guided policy search to sensorimotor deep learning. Guided policy search decomposes the policy learning problem into two phases: a trajectory optimization phase that determines how to solve the task in a few specific conditions, and a supervised learning phase that trains the policy from these successful executions with supervised learning [22]. Since the CNN policy is trained with supervised learning, we can use the tools developed in the deep learning community to make this phase simple and efficient. We handle the partial observability of visuomotor control by optimizing the trajectories with full state information, while providing only partial observations (consisting of images and robot configurations) to the policy. The trajectories are optimized under unknown dynamics, using real-world experience and minimal prior knowledge.

The main contribution of our work is a method for end-to-end training of deep visuomotor policies for robotic manipulation. We propose a partially observed guided policy search algorithm that can train high-dimensional policies for tasks where part of the state must be determined from camera images. We also introduce a novel CNN architecture designed for robotic control, shown in Figure 2. The vision layers of this CNN are designed for localizing points of interest in an image, unlike standard vision architectures that discard locational information to induce translational invariance [16]. We evaluate our method by learning policies for inserting a block into a shape sorting cube, screwing a cap onto a bottle,

Technical report published 2/22/2015 as BVLC Report Series #100.
Asterisk (*) denotes authors who contributed equally to the work.

fitting the claw of a toy hammer under a nail with various grasps, and placing a coat hanger on a rack (see Figure 1). Our results demonstrate clear improvements in consistency and generalization from training visuomotor policies end-to-end, when compared to using the poses or features produced by a CNN trained for 3D object localization.

II. Related Work

Reinforcement learning and policy search have been applied in robotics for playing games such as table tennis [13], object manipulation [2, 29], and locomotion [5, 7, 15, 39]. Several recent papers provide surveys of policy search in robotics [3, 14]. Such methods are typically applied to one component of the robot control pipeline, which often sits on top of a hand-designed controller, such as a PD controller, and accepts processed input, for example from an existing vision pipeline [29]. Our method trains policies that map visual input and joint encoder signals directly to the torques at the robot's joints. By learning the entire mapping from perception to control, the perception layers can be adapted to optimize task performance.

We represent our policies with convolutional neural networks (CNNs). CNNs have recently achieved dramatic improvements on a number of vision benchmarks [8, 16, 40]. Most applications of CNNs focus on classification, where locational information is intentionally discarded by means of successive pooling layers [18]. Applications to localization typically either use a sliding window to localize the object, reducing the task to classification [8], perform regression to a heatmap of manually labeled keypoints [40], requiring precise knowledge of the object position in the image and camera calibration, or use 3D models to localize previously scanned objects [30, 36]. We use a novel CNN architecture that automatically learns feature points without any supervision beyond the information from the robot's encoders and camera.

Unlike with visual recognition, applications of deep networks to robotic control have been comparatively limited. Backpropagation through the dynamics and the image formation process is impractical, since they are often non-differentiable, and such long-range backpropagation leads to extreme numerical instability. The high dimensionality of the network also makes reinforcement learning very difficult [3]. Pioneering early work on neural network control used small, simple networks [10, 33], and has largely been supplanted by methods that use carefully designed policies that can be learned efficiently with reinforcement learning [14]. More recent work on sensorimotor deep learning has tackled simple task-space motion [17] and used unsupervised learning to obtain low-dimensional state spaces from images [34], but such methods are limited to tasks with a low-dimensional structure. CNNs have also been trained to play video games with temporal difference learning and Monte Carlo tree search [9, 26]. However, such methods have only been demonstrated on discrete, synthetic domains, and require an impractical number of samples for real-world robotic applications. Our method is sample efficient, requiring only minutes of interaction time. To the best of our knowledge, this is the first method that can train deep visuomotor policies for complex, high-dimensional manipulation skills with direct torque control.

Learning visuomotor policies end-to-end introduces two key challenges: partial observability and the high dimensionality of the policy. We tackle these challenges using guided policy search. In guided policy search, the policy is optimized using supervised learning, which scales gracefully with the dimensionality of the function approximator. The training set for this supervised learning procedure can be constructed from example demonstrations [20], trajectory optimization under known dynamics [21, 22, 28], and trajectory-centric reinforcement learning methods that operate under unknown dynamics [19, 23], which is the approach taken in this work. We propose a new, partially observed guided policy search method based on the Bregman alternating directions method of multipliers (BADMM) that makes it practical to train complex, generalizable policies under partial observation.

The goal of our approach is also similar to visual servoing, which performs feedback control on feature points in a camera image [6, 27, 42]. However, our visuomotor policies are entirely learned from real-world data, and do not require feature points or feedback controllers to be specified by hand. This gives our method considerable flexibility in choosing how to use the visual signal. Furthermore, our approach does not require any sort of camera calibration, in contrast to many visual servoing methods (though not all – see e.g. [11, 43]).

III. Overview

The aim of our method is to learn a policy $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ that specifies a distribution over actions $\mathbf{u}_t$ conditioned on the observation $\mathbf{o}_t$, which includes a camera image and the configuration of the robot. The policy parameters θ are optimized to minimize a cost function $\ell(\mathbf{x}_t, \mathbf{u}_t)$ over the course of a fixed-length episode. The actions $\mathbf{u}_t$ are the motor torques, and the state $\mathbf{x}_t$ includes the known robot configuration as well as (for example) the target position for an object placement task or the grasp pose. The latter information is not observed directly by the policy, and must be inferred from the camera image. We represent $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ as a Gaussian, with the mean given by a nonlinear function approximator. Since this function approximator needs to operate directly on raw images, we use convolutional neural networks (CNNs), which have enjoyed considerable success in computer vision [16]. The architecture of our CNN is shown in Figure 2. This network has 7 layers and around 92,000 parameters, which presents a tremendous challenge for standard policy search methods [3].

To handle this high dimensionality and the challenge of partial observability, we extend the framework of guided policy search. In guided policy search, the policy is trained with supervised learning, which scales well even to very high-dimensional function approximators. To construct the training set for supervised learning, we employ a trajectory-centric reinforcement learning method that finds good trajectories from a number of initial states. This phase does not require knowledge of the system dynamics, but does use the full state $\mathbf{x}_t$, which makes learning very efficient. For example,

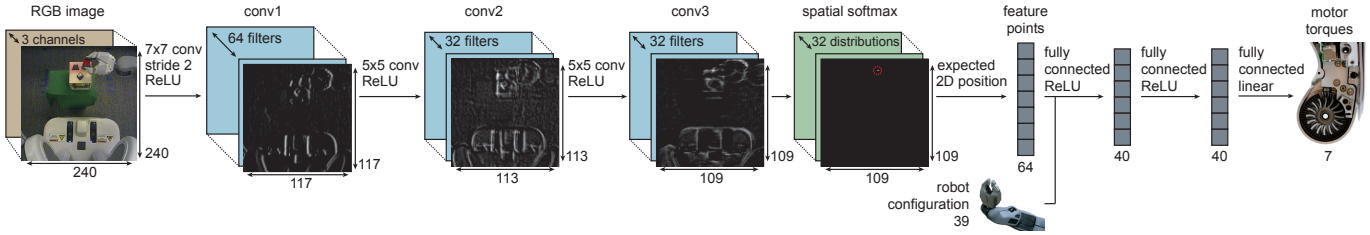


Fig. 2: Visuomotor policy architecture. The network contains three convolutional layers, followed by a spatial softmax and an expected position layer that converts pixel-wise features to feature points, which are better suited for spatial computations. The points are concatenated with the robot configuration, then passed through three fully connected layers to produce the torques.

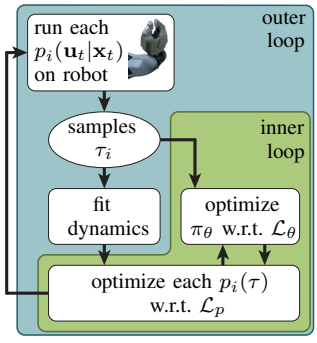
if the unobserved part of $\mathbf{x}_t$ is the position of a target object, such as the bottle, we can hold this object in the robot's left gripper, while the right arm performs the task. This type of instrumented training is a natural fit for many robotic tasks, where the training is performed in a controlled environment, but the final policy must be able to succeed "in the wild."

Naïve supervised learning will often fail to produce a good policy, since a small mistake on the part of the policy will put it in states that are not part of the training, causing compounding errors. To avoid this problem, the training data must come from the policy's own state distribution [35]. We use BADMM [41] to adapt the trajectories to the policy, alternating between optimizing the policy to match the trajectories, and optimizing the trajectories to minimize cost and match the policy, such that at convergence, they have the same state distribution.

IV. Partially Observed Guided Policy Search

Guided policy search methods transform policy search into a supervised learning problem, where the training set is generated by simple trajectory-centric algorithms. The trajectory phase produces Gaussian trajectory distributions $p_i(\tau)$, which correspond to a mean trajectory with linear feedback. Each $p_i(\tau)$ succeeds from a specific initial state. For example, in the task of placing a cap on a bottle, these initial states corresponds to the positions of the bottle. By training on multiple trajectories for multiple bottle positions, the final CNN policy can succeed from all initial states, and can generalize to other states from the same distribution.

We present a partially observed guided policy search method that uses BADMM to iteratively enforce agreement between the policy $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ and the trajectory distributions $p_i(\tau)$. A diagram of this method is shown on the right. In the outer loop, we draw a sample for each initial state on the real system. The samples are used to fit the dynamics for trajectory optimization, and serve as training data for the policy. The inner loop alternates between optimizing each $p_i(\tau)$ and optimizing the policy. Unlike prior guided policy search methods, the policy is trained on observations $\mathbf{o}_t$, allowing the method to handle partial observability, while the trajectories are optimized on the full state $\mathbf{x}_t$. Using BADMM allows us to formulate simple and efficient optimizations for both inner loop phases. We derive this algorithm below, followed by a discussion of the two inner loop phases and a comparison with prior guided policy search methods.



A. Algorithm Derivation

Policy search methods minimize the expected cost $E_{\pi_\theta}[\ell(\tau)]$ of the policy π_θ, where $\tau = \{\mathbf{x}_1, \mathbf{u}_t, \ldots, \mathbf{x}_T, \mathbf{u}_T\}$ is a trajectory, and $\ell(\tau) = \sum_{t=1}^{T} \ell(\mathbf{x}_t, \mathbf{u}_t)$ is the cost of an episode. In the fully observed case, the expectation is taken under $\pi_\theta(\tau) = p(\mathbf{x}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{u}_t|\mathbf{x}_t) p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$. For a partially observed task, we only know $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$, but $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$ can be recovered as $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t) = \int \pi_\theta(\mathbf{u}_t|\mathbf{o}_t) p(\mathbf{o}_t|\mathbf{x}_t) d\mathbf{o}_t$. We will present the derivation in this section for $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$, but we do not require knowledge of $p(\mathbf{o}_t|\mathbf{x}_t)$ in the final algorithm. As discussed in Section IV-C, the integral will be evaluated with samples from the real system. We begin by rewriting the expected cost minimization as a constrained problem:

$$\min_{p, \pi_\theta} E_p[\ell(\tau)] \text{ s.t. } p(\mathbf{u}_t|\mathbf{x}_t) = \pi_\theta(\mathbf{u}_t|\mathbf{x}_t) \; \forall \, \mathbf{x}_t, \mathbf{u}_t, t, \qquad (1)$$

where $p(\tau)$ is another distribution. This formulation is equivalent to the original problem, since the constraint forces the two distributions to be identical. However, if we approximate the initial state distribution $p(\mathbf{x}_1)$ with samples $\mathbf{x}_1^i$, we can choose $p(\tau)$ to be a class of distributions that is much easier to optimize than π_θ, as we will show later. The constrained problem can be solved by a dual descent method, which alternates between minimizing the Lagrangian with respect to the primal variables, and incrementing the Lagrange multipliers by their subgradient. Minimization of the Lagrangian with respect to $p(\tau)$ and θ is done in alternating fashion: minimizing with respect to θ corresponds to supervised learning (making π_θ match $p(\tau)$), and minimizing with respect to $p(\tau)$ consists of one or more trajectory optimization problems. The dual descent method we use is based on BADMM [41], a variant of ADMM that augments the Lagrangian with a Bregman divergence between the constrained variables. We use the KL-divergence and replace the constraint with the equivalent constraint $p(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t) = \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t)$, which yields

iterations with the following steps:

$$\theta \leftarrow \arg\min_{\theta} \sum_{t=1}^{T} E_{p(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}[\lambda_{\mathbf{x}_t,\mathbf{u}_t}] + \nu_t \phi_t^\theta(\theta, p)$$

$$p \leftarrow \arg\min_{p} \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\ell(\mathbf{x}_t, \mathbf{u}_t) - \lambda_{\mathbf{x}_t,\mathbf{u}_t}] + \nu_t \phi_t^p(p, \theta)$$

$$\lambda_{\mathbf{x}_t,\mathbf{u}_t} \leftarrow \alpha\nu_t(\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t) - p(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t)),$$

where $\lambda_{\mathbf{x}_t,\mathbf{u}_t}$ is the Lagrange multiplier for state $\mathbf{x}_t$ and action $\mathbf{u}_t$ at time t, $\phi_t^p(p,\theta) = E_{p(\mathbf{x}_t)}[D_{\mathrm{KL}}(p(\mathbf{u}_t|\mathbf{x}_t)\|\pi_\theta(\mathbf{u}_t|\mathbf{x}_t))]$ and $\phi_t^\theta(\theta, p) = E_{p(\mathbf{x}_t)}[D_{\mathrm{KL}}(\pi_\theta(\mathbf{u}_t|\mathbf{x}_t))\|p(\mathbf{u}_t|\mathbf{x}_t)]$ are the KL-divergence terms, and α is a step size.

The dynamics only affect the optimization with respect to $p(\tau)$. In order to make this optimization efficient, we choose $p(\tau)$ to be a mixture of N Gaussians $p_i(\tau)$, one for each initial state sample $\mathbf{x}_1^i$. This makes the action conditionals $p_i(\mathbf{u}_t|\mathbf{x}_t)$ and the dynamics $p_i(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ linear Gaussian. This is a reasonable choice when the system is deterministic, or the noise is Gaussian or small, and we found that this approach is sufficiently tolerant to noise for use on real physical systems. Our choice of p also assumes that the policy $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ is conditionally Gaussian. This is also reasonable, since the mean and covariance of $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ can be any nonlinear function of the observations $\mathbf{o}_t$, which themselves are a function of the unobserved state $\mathbf{x}_t$. In Section IV-B, we show how these assumptions enable each $p_i(\tau)$ to be optimized very efficiently.

But first, we must choose a tractable way to represent the infinite set of constraints $p(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t) = \pi_\theta(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t)$. One approach proposed in prior work on policy search for approximating such infinite constraints is to replace them with expectations of features [32]. When the features consist of linear, quadratic, or higher order monomial functions of the random variable, this can be viewed as a constraint on the moments of the distributions. If we only use the first moment, we get a constraint on the expected action: $E_{p(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t)}[\mathbf{u}_t] = E_{\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t)}[\mathbf{u}_t]$. If the stochasticity in the dynamics is low, as we assumed previously, the optimal solution for each $p_i(\tau)$ will have low entropy, making this first moment constraint a reasonable approximation. Furthermore, the KL-divergence terms in the augmented Lagrangians will still serve to softly enforce agreement between the higher moments. While this simplification is quite drastic, we found that it was more stable in practice than including higher moments. The alternating optimization is now given by

$$\theta \leftarrow \arg\min_{\theta} \sum_{t=1}^{T} E_{p(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}[\mathbf{u}_t^{\mathrm{T}} \lambda_{\mu t}] + \nu_t \phi_t^\theta(\theta, p)$$

$$p \leftarrow \arg\min_{p} \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\ell(\mathbf{x}_t, \mathbf{u}_t) - \mathbf{u}_t^{\mathrm{T}} \lambda_{\mu t}] + \nu_t \phi_t^p(p, \theta)$$

$$\lambda_{\mu t} \leftarrow \alpha\nu_t(E_{\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t)}[\mathbf{u}_t] - E_{p(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t)}[\mathbf{u}_t]),$$

where $\lambda_{\mu t}$ is the Lagrange multiplier on the expected action at time t. In the algorithm diagram, we use $\mathcal{L}_\theta(\theta, p)$ and $\mathcal{L}_p(p, \theta)$ as shorthand for the two augmented Lagrangians minimized with respect to θ and p, respectively. In the next two sections, we will describe how $\mathcal{L}_p(p, \theta)$ can be optimized with respect to p under unknown dynamics, and how $\mathcal{L}_\theta(\theta, p)$ can be optimized for complex, high-dimensional policies. Implementation details of the BADMM optimization are presented in the supplementary appendix.

B. Trajectory Optimization under Unknown Dynamics

Since the Lagrangian $\mathcal{L}_p(p, \theta)$ in the previous section factorizes over the mixture elements in $p(\tau) = \sum_i p_i(\tau)$, we describe the trajectory optimization method for a single Gaussian $p(\tau)$. When there are multiple mixture elements, this procedure is applied in parallel to each $p_i(\tau)$. The derivation follows prior work [19], and we describe novel modifications for improving performance and sample efficiency at the end.

Since $p(\tau)$ is Gaussian, the conditionals $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ and $p(\mathbf{u}_t|\mathbf{x}_t)$, which correspond to the dynamics and the controller, are linear-Gaussian. The dynamics are determined by the environment. If the dynamics are known, $p(\mathbf{u}_t|\mathbf{x}_t)$ can be optimized with a variant of the iterative linear-quadratic regulator [20, 24]. In the case of unknown dynamics, we can fit $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ to sample trajectories gathered from the trajectory distribution at the previous iteration, denoted $\hat{p}(\tau)$. If $\hat{p}(\tau)$ is too different from $p(\tau)$, these samples will not give a good estimate of $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$, and the optimization will diverge. To avoid this, we can bound the change from $\hat{p}(\tau)$ to $p(\tau)$ in terms of their KL-divergence by a step size ϵ, resulting in the following constrained problem:

$$\min_{p(\tau)\in\mathcal{N}(\tau)} \mathcal{L}_p(p, \theta) \text{ s.t. } D_{\mathrm{KL}}(p(\tau)\|\hat{p}(\tau)) \leq \epsilon.$$

This type of policy update has previously been proposed by several authors in the context of policy search [1, 19, 31, 32]. In the case when $p(\tau)$ is Gaussian, this problem can be solved efficiently using dual gradient descent, while the dynamics $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ are fitted to samples gathered by running the previous controller $\hat{p}(\mathbf{u}_t|\mathbf{x}_t)$ on the robot. Fitting a global Gaussian mixture model to tuples $(\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1})$ and using it as a prior for fitting the dynamics $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ serves to greatly reduce the sample complexity. Note that this constrained optimization is performed in the "inner loop" of the optimization described in the previous section. The overall algorithm then becomes an instance of generalized BADMM [41].

Note that the augmented Lagrangian $\mathcal{L}_p(p, \theta)$ consists of an expectation under $p(\tau)$ of a quantity that is independent of p. We can locally approximate this quantity with a quadratic by using a quadratic expansion of $\ell(\mathbf{x}_t, \mathbf{u}_t)$, and fitting a linear-Gaussian to $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$ with the same method we used for the dynamics. We can then solve the primal optimization in the dual gradient descent procedure with a standard LQR backward pass. This is significantly simpler and much faster than the forward-backward dynamic programming procedure employed in previous work [19, 22]. This improvement is enabled by the use of BADMM, which allows us to always formulate the KL-divergence term in the Lagrangian with the distribution being optimized as the first argument. Since the KL-divergence is convex in its first argument, this makes the corresponding optimization significant easier.

We also depart from previous work by allowing samples from multiple trajectories $p_i(\tau)$ to be used to fit a shared dynamics $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$, while the controllers $p_i(\mathbf{u}_t|\mathbf{x}_t)$ are allowed to vary. This makes sense when the initial states of these trajectories are similar, and they therefore visit similar regions. This allows us to draw just a single sample from each $p_i(\tau)$ at each iteration, allowing us to handle many more initial states. Prior work required around 5 samples per initial state, and was therefore limited to only a few initial states [23].

C. Supervised Policy Optimization

Since the policy parameters θ participate only in the constraints of the optimization problem in Equation 1, optimizing the policy corresponds to minimizing the KL-divergence between the policy and trajectory distribution, as well as the expectation of $\lambda_{\mu t}^{\mathrm{T}}\mathbf{u}_t$. For a conditional Gaussian policy of the form $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t) = \mathcal{N}(\mu^\pi(\mathbf{o}_t), \Sigma^\pi(\mathbf{o}_t))$, the objective is

$$\mathcal{L}_\theta(\theta, p) = \frac{1}{2N}\sum_{i=1}^{N}\sum_{t=1}^{T} E_{p_i(\mathbf{x}_t,\mathbf{o}_t)}\big[\mathrm{tr}[\mathbf{C}_{ti}^{-1}\Sigma^\pi(\mathbf{o}_t)] - \log|\Sigma^\pi(\mathbf{o}_t)| + (\mu^\pi(\mathbf{o}_t) - \mu_{ti}^p(\mathbf{x}_t))\mathbf{C}_{ti}^{-1}(\mu^\pi(\mathbf{o}_t) - \mu_{ti}^p(\mathbf{x}_t)) + 2\lambda_{\mu t}^{\mathrm{T}}\mu^\pi(\mathbf{o}_t)\big],$$

where $\mu_{ti}^p(\mathbf{x}_t)$ is the mean of $p_i(\mathbf{u}_t|\mathbf{x}_t)$ and $\mathbf{C}_{ti}$ is the covariance, and the expectation is evaluated using samples from each $p_i(\tau)$ with corresponding observations $\mathbf{o}_t$. The observations are sampled from $p(\mathbf{o}_t|\mathbf{x}_t)$ by recording camera images on the real system. Since the input to $\mu^\pi(\mathbf{o}_t)$ and $\Sigma^\pi(\mathbf{o}_t)$ is not the state $\mathbf{x}_t$, but only an observation $\mathbf{o}_t$, we can train the policy for partially observed tasks. Note that $\mathcal{L}_\theta(\theta, p)$ is simply a weighted quadratic loss on the difference between the policy mean and the mean action of the trajectory distribution, offset by the Lagrange multiplier. The weighting is the precision matrix of the conditional in the trajectory distribution, which is equal to the curvature of its cost-to-go function [20]. This has an intuitive interpretation: $\mathcal{L}_\theta(\theta, p)$ penalizes deviation from the trajectory distribution, with a penalty that is proportional to the approximate cost-to-go function.

We optimize $\mathcal{L}_\theta(\theta, p)$ with respect to θ using stochastic gradient descent (SGD), a standard method for neural network training. The covariance of the Gaussian policy does not depend on the observation in our prototype, though adding this dependence would be straightforward. Since training complex neural networks requires a substantial number of samples, we found it beneficial to include sampled observations from previous iterations into the policy optimization, evaluating the action $\mu_{ti}^p(\mathbf{x}_t)$ at their corresponding states using the current trajectory distributions. Since these samples come from the wrong state distribution, we use importance sampling and weight them according to the ratio of their probability under the current distribution $p(\mathbf{x}_t)$ and the one they were sampled from, which is straightforward to evaluate under linear-Gaussian dynamics [21].

D. Comparison with Prior Guided Policy Search Methods

We presented the first guided policy search method where the policy is trained on observations, while the trajectories are trained on the full state. The BADMM formulation of guided policy search is also new to this work, though several prior guided policy search methods based on constrained optimization have been proposed. Levine et al. proposed a formulation similar to Equation 1, but with a constraint on the KL-divergence between $p(\tau)$ and π_θ [22]. This results in a more complex forward-backward trajectory optimization phase. We found that our method was more effective at finding a good policy in fewer iterations, due to the Lagrange multipliers on the actions that more aggressively correct discrepancies between the policy and trajectories. We also found our method to be much more computationally efficient, especially when the number of trajectories $p_i(\tau)$ is large, since the trajectories are optimized with standard LQR backward passes.

The use of ADMM for guided policy search was also proposed by Mordatch et al. for deterministic policies under known dynamics [28]. This approach requires known, deterministic dynamics and trains deterministic policies. Furthermore, because this approach uses a simple quadratic augmented Lagrangian term, it further requires penalty terms on the gradient of the policy to account for local feedback. Our approach enforces this feedback behavior due to the higher moments included in the KL-divergence term, but does not require computing the second derivative of the policy.

V. End-to-End Visuomotor Policies

Guided policy search allows us to optimize complex, high-dimensional policies that act under partial observation. In this section, we describe a policy architecture that uses images from a monocular camera to execute a variety of manipulation tasks, as well as our training procedure.

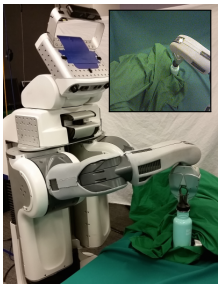
A. Visuomotor Policy Architecture

Our visuomotor policy runs at 20 Hz on the robot, mapping monocular RGB images and the robot configurations to joint torques on a 7 DoF arm. The configuration includes the angles of the joints and the pose of the end-effector (defined by 3 points), as well as their velocities, but does not include the position of the target object or goal, which must be determined from the image. CNNs often use pooling to discard the locational information that is necessary to determine positions, since it is an irrelevant distractor for tasks such as object classification [18]. Because locational information is important for control, our policy does not use pooling. Additionally, CNNs built for spatial tasks such as human pose estimation often also rely on the availability of location labels in image-space, such as hand-labeled keypoints [40]. We propose a novel CNN architecture capable of estimating spatial information from an image without direct supervision in image space. Our pose estimation experiments, discussed in Section V-B, show that this network can learn useful visual features using only 3D positional information provided by the robot and no camera calibration. Furthermore, by training our network with guided policy search, it can acquire *task-specific* visual features that improve policy performance.

Our network architecture is shown in Figure 2. The visual processing layers of the network consist of three convolutional layers, each of which learns a bank of filters that are applied to patches centered on every pixel of its input. These filters form a hierarchy of local image features. Each convolutional layer is followed by a rectifying nonlinearity of the form $a_{cij} = max(0, z_{cij})$ for each channel c and each pixel coordinate (i, j). The third convolutional layer contains 32 response maps with resolution 109×109. These response maps are passed through a spatial softmax function of the form $s_{cij} = e^{a_{cij}} / \sum_{i'j'} e^{a_{ci'j'}}$. Each output channel of the softmax is a probability distribution over the location of a feature in the image. To convert from this distribution to a spatial representation, the network calculates the expected image position of each feature, yielding a 2D coordinate for each channel. These feature points are concatenated with the robot's configuration and fed through two fully connected layers, each with 40 rectified units, followed by linear connections to the torques. The full visuomotor policy contains about 92,000 parameters, of which 86,000 are in the convolutional layers.

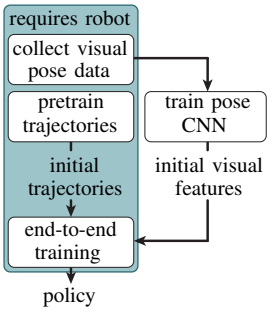
The spatial softmax and the expected position computation serve to convert pixel-wise representations in the convolutional layers to spatial coordinate representations, which can be manipulated by the fully connected layers into 3D positions or motor torques. The softmax also provides lateral inhibition, which suppresses low, erroneous activations. This makes our policy more robust to distractors, providing generalization to novel visual variation. We compare our architecture with more standard alternatives in Section VI-C.

B. Visuomotor Policy Training



We train the policy using the full state during the trajectory optimization phase, though the final policy acts under partial observations. This type of instrumented training is a natural choice for many robotics tasks, where the robot is trained under controlled conditions, but must then act intelligently in uncontrolled, real-world situations. In our tasks, the unobserved variables are the pose of a target object (e.g. the bottle on which a cap must be placed). During training, this target object is held in the robot's left gripper, while the robot's right arm performs the task, as shown above. This allows the robot to move the target through a range of known positions. The final visuomotor policy does not receive this position as input, but must instead use the camera images. The left arm is covered with cloth to prevent the policy from associating its appearance with the object's position.

While we can train the visuomotor policy entirely on the robot, the algorithm would spend a large number of iterations learning basic visual features and arm motions that can more efficiently be learned by themselves, before being incorporated into the policy. To speed up learning, we initialize both the vision layers in the policy and the trajectory distributions for guided policy search by leveraging the fully observed training setup. This initialization does not use any additional information that is not already available from the robot. To initialize the vision layers, the robot moves the target object through a range of random positions, recording the object's pose and camera images. This dataset is used to train a pose regression CNN, which consists of the same vision layers as the policy, followed by a fully connected layer that outputs the 3D points that define the target. Since the training set is still quite limited, we initialize the filters in the first layer with weights from the model of Szegedy et al. [38], which is trained on ImageNet [4] classification. After training on pose regression, the weights in the convolutional layers are transferred to the policy CNN. This enables the robot to learn the appearance of the objects prior to learning the behavior.

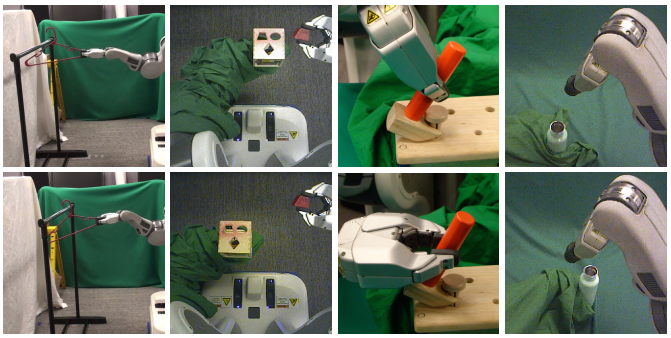


To initialize the trajectories, we take 15 iterations of guided policy search without optimizing the visuomotor policy. This allows for much faster training in the early iterations, when the trajectories are not yet successful, and optimizing the full visuomotor policy is unnecessarily time consuming. Since we still want the trajectories to arrive at compatible strategies for each target position, we replace the visuomotor policy during these iterations with a small network that receives the full state. This network serves only to constrain the trajectories and avoid divergent behaviors from emerging for similar initial states, which would make subsequent policy learning difficult. As shown in the diagram above, the trajectories can be pre-trained in parallel with the vision layer pre-training, which does not require the robot.

After initialization, we train the full visuomotor policy with guided policy search. During the supervised policy optimization phase, the fully connected motor control layers are first optimized by themselves, since they are not initialized with pre-training. Then, the entire network is further optimized end-to-end. We found that this setup prevents the convolutional layers from forgetting the useful features learned during pre-training while still training all layers of the policy.

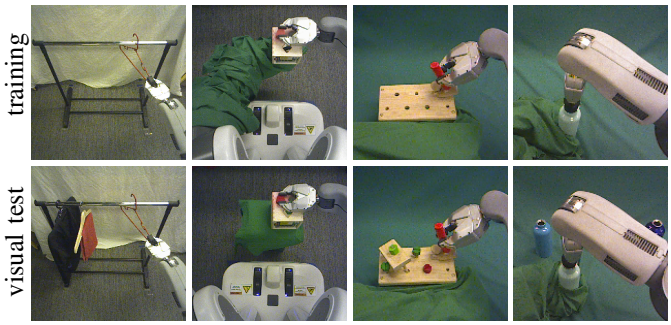
VI. Experimental Results

We evaluated our method by training policies for hanging a coat hanger on a clothes rack, inserting a block into a shape sorting cube, fitting the claw of a toy hammer under a nail with various grasps, and screwing on a bottle cap. The cost function for these tasks encourages low distance between three points on the end-effector and corresponding target points, low torques, and, for the bottle task, spinning the wrist. The equations for these cost functions follow prior work [23]. The tasks are illustrated in Figure 3. Each task involved variation of about 10-20 cm in each direction in the position of the target object (the rack, shape sorting cube, nail, and bottle). In addition, the coat hanger and hammer tasks were trained with two and three grasps, respectively. All tasks used the same policy architecture and model parameters.



(a) hanger (b) cube (c) hammer (d) bottle

Fig. 3: Illustration of the tasks in our experiments, showing the variation in the position of the target for the hanger, cube, and bottle tasks, as well as two of the three grasps for the hammer, which also included variation in position (not shown).



(a) hanger (b) cube (c) hammer (d) bottle

Fig. 4: Training and visual test scenes as seen by the policy at the ends of successful episodes. The hammer and bottle images were cropped for visualization only.

A. *Visuomotor Policy Generalization*

We evaluated the visuomotor policies in three conditions: (1) the training target positions and grasps, (2) new target positions not seen during training and, for the hammer, new grasps (spatial test), and (3) training positions with visual distractors (visual test). A selection of these experiments is shown in the supplementary video. For the visual test, the shape sorting cube was placed on a table rather than held in the gripper, the coat hanger was placed on a rack with clothes, and the bottle and hammer tasks were done in the presence of clutter. Illustrations of this test are shown in Figure 4.

The success rates for each test are shown in Table I. We compared to two baselines, both of which train the vision layers in advance for pose prediction, instead of training the entire policy end-to-end. The features baseline discards the last layer of the pose predictor and uses the feature points, resulting in the same architecture as our policy, while the prediction baseline feeds the predicted pose into the control layers.

The pose prediction baseline is analogous to a standard modular approach to policy learning, where the vision system is first trained to localize the target, and the policy is trained on top of it. This variant achieves poor performance, because although the pose is accurate to about 1 cm, this is insufficient for such precise tasks. As shown in the video, the shape sorting cube and bottle cap insertions have tolerances of just a few millimeters. Such accuracy is difficult to achieve even with calibrated cameras and checkerboards. Indeed, prior work has reported that the PR2 can maintain a camera to end effector accuracy of about 2 cm during open loop motion [25]. This suggests that the failure of this baseline is not atypical, and that our visuomotor policies are learning visual features and control strategies that improve the robot's accuracy.

When provided with pose estimation features, the policy has more freedom in how it uses the visual information, and achieves somewhat higher success rates. However, full end-to-end training performs significantly better, achieving high accuracy even on the challenging bottle task, and successfully adapting to the variety of grasps on the hammer task. This suggests that, although the vision layer pre-training is clearly beneficial for reducing computation time, it is not sufficient by itself for discovering good features for visuomotor policies.

coat hanger	training (18)	spatial test (24)	visual test (18)
end-to-end training	**100%**	**100%**	**100%**
pose features	88.9%	87.5%	83.3%
pose prediction	55.6%	58.3%	66.7%
shape sorting cube	training (27)	spatial test (36)	visual test (40)
end-to-end training	**96.3%**	**91.7%**	**87.5%**
pose features	70.4%	83.3%	40%
pose prediction	0%	0%	n/a
toy claw hammer	training (45)	spatial test (60)	visual test (60)
end-to-end training	**91.1%**	**86.7%**	**78.3%**
pose features	62.2%	75.0%	53.3%
pose prediction	8.9%	18.3%	n/a
bottle cap	training (27)	spatial test (12)	visual test (40)
end-to-end training	**88.9%**	**83.3%**	**62.5%**
pose features	55.6%	58.3%	27.5%

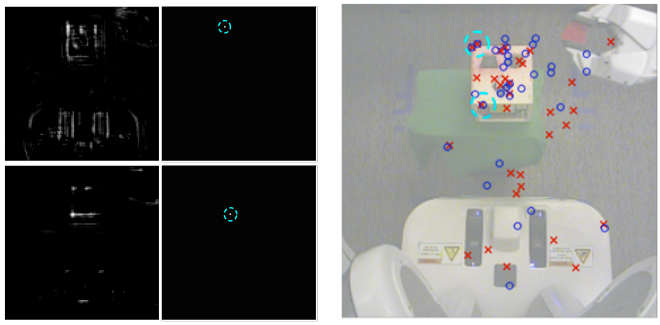
TABLE I: Success rates on training positions, on novel test positions, and in the presence of visual distractors. The number of trials per test is shown in parentheses.

The policies exhibit moderate tolerance to distractors that are visually separated from the target object. However, as expected, they tend to perform poorly under drastic changes to the backdrop, or when the distractors are adjacent to or occluding the manipulated objects, as shown in the supplementary video. In future work, this could be mitigated by varying the scene at training time, or by artificially augmenting the image samples with synthetic transformations, as discussed in prior work in computer vision [37].

B. *Features Learned with End-to-End Training*

In Figure 5, we compare the feature points learned through guided policy search to those learned by a CNN trained for pose prediction. After end-to-end training, the policy acquired a distinctly different set of feature points compared to the pose prediction CNN used for initialization. The end-to-end trained model finds more feature points on task-relevant objects and fewer points on background objects. This suggests

The video can be viewed at http://sites.google.com/site/visuomotorpolicy



(a) conv3 (b) softmax (c) feature points

Fig. 5: Feature points learned by the shape sorting cube policy. Two of the 32 conv3 response maps are shown in (a), and the corresponding softmax distributions are displayed in (b). In (c), we show the output feature points for this input image in blue, while the feature points of the pose prediction network are shown in red. The end-to-end trained model discovers more feature points on the cube and the gripper.

that the policy improves its performance by acquiring *task-specific* visual features that differ from those learned for object localization. We further analyze the features learned by our policies in the supplementary appendix.

C. CNN Architecture Evaluation

To evaluate the visual processing portion of our architecture, we measured its accuracy on the pose estimation pre-training task discussed in Section V-B. We compare to a network where the fixed transformation from the softmax to the feature points is replaced with a conventional learned fully connected layer, as well as to networks that omit the softmax and use 3×3 max pooling with stride 2 at the first two layers. These alternative architectures have many more parameters, since the new fully connected layer takes as input the entire bank of response maps from the third convolutional layer. The results in Table II indicate that using the softmax and the fixed transformation from the softmax output to the spatial feature representation improves pose estimation accuracy and reduces overfitting. Our network is able to outperform the more standard architectures because it is forced by the softmax and expected position layers to learn feature points, which provide a concise representation suitable for spatial inference. The lower number of parameters also results in an easier optimization and reduces overfitting.

network architecture	training error (cm)	test error (cm)
softmax + feature points (**ours**)	**1.14 ± 1.67**	**1.30 ± 0.73**
softmax + fully connected layer	2.27 ± 1.70	2.59 ± 1.19
fully connected layer	4.65 ± 2.90	4.75 ± 2.29
max-pooling + fully connected	2.89 ± 2.08	3.71 ± 1.73

TABLE II: Average pose estimation accuracy and standard deviation with various architectures, measured as average Euclidean error for the three target points in 3D, with ground truth determined by forward kinematics from the left arm.

D. Implementation and Computational Performance

CNN training was implemented using the Caffe [12] deep learning library. Each visuomotor policy required 3-4 hours of training time: 20-30 minutes for the pose prediction data collection on the robot, 40-60 minutes for the fully observed trajectory pre-training on the robot and offline pose pre-training (which can be done in parallel), and between 1.5 and 2.5 hours for end-to-end training with guided policy search. The coat hanger task required two iterations of guided policy search, the shape sorting cube and the hammer required three, and the bottle task required four. Training time was dominated by computation rather than robot interaction time, and we expect significant speedup from a more efficient implementation.

VII. Discussion and Future Work

In this paper, we presented a method for learning robotic control policies that use raw input from a monocular camera. These policies are represented by a novel convolutional neural network architecture, and can be trained end-to-end using our partially observed guided policy search algorithm, which decomposes the policy search problem in a trajectory optimization phase that uses full state information and a supervised learning phase that only uses partial observations. This decomposition allows us to leverage state-of-the-art tools from supervised learning, making it straightforward to optimize extremely high-dimensional policies. Our experimental results show that our method can execute complex manipulation skills, and that end-to-end training produces significant improvements in policy performance compared to using fixed vision layers trained for pose prediction.

Although we demonstrate moderate generalization over variations in the scene, our current method does not generalize to dramatically different settings, especially when visual distractors occlude the manipulated object or break up its silhouette in ways that differ from the training. The success of CNNs on exceedingly challenging vision tasks suggests that this class of models is capable of learning invariance to irrelevant distractor features [8, 16, 40], and in principle this issue can be addressed by training the policy in a variety of environments, though this poses certain logistical challenges. More practical alternatives that could be explored in future work include simultaneously training the policy on multiple robots, each of which is located in a different environment, developing more sophisticated regularization and pre-training techniques to avoid overfitting, and introducing artificial data augmentation to encourage the policy to be invariant to irrelevant clutter. However, even without these improvements, our method has numerous applications in, for example, an industrial setting where the robot must repeatedly and efficiently perform a task that requires visual feedback under moderate variation in background and clutter conditions.

In future work, we hope to explore more complex policy architectures, such as recurrent policies that can deal with extensive occlusions by keeping a memory of past observations. We also hope to extend our method to a wider range

of tasks that can benefit from visual input, as well as a variety of other rich sensory modalities, including haptic input from pressure sensors and auditory input. With a wider range of sensory modalities, end-to-end training of sensorimotor policies will become increasingly important: while it is often straightforward to imagine how vision might help to localize the position of an object in the scene, it is much less apparent how sound can be integrated into robotic control. A learned sensorimotor policy would be able to naturally integrate a wide range of modalities and utilize them to directly aid in control.

REFERENCES

[1] J. A. Bagnell and J. Schneider. Covariant policy search. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[2] M. Deisenroth, C. Rasmussen, and D. Fox. Learning to control a low-cost manipulator using data-efficient reinforcement learning. In *Robotics: Science and Systems (RSS)*, 2011.

[3] M. Deisenroth, G. Neumann, and J. Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2013.

[4] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[5] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng. Learning CPG-based biped locomotion with a policy gradient method: Application to a humanoid robot. *International Journal of Robotic Research*, 27(2):213–228, 2008.

[6] B. Espiau, F. Chaumette, and P. Rives. A new approach to visual servoing in robotics. *IEEE Transactions on Robotics and Automation*, 8(3), 1992.

[7] T. Geng, B. Porr, and F. Wörgötter. Fast biped walking with a reflexive controller and realtime policy searching. In *Advances in Neural Information Processing Systems (NIPS)*, 2006.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[9] X. Guo, S. Singh, H. Lee, R. L. Lewis, and X. Wang. Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[10] K. J. Hunt, D. Sbarbaro, R. Żbikowski, and P. J. Gawthrop. Neural networks for control systems: A survey. *Automatica*, 28(6):1083–1112, November 1992.

[11] M. Jägersand, O. Fuentes, and R. C. Nelson. Experimental evaluation of uncalibrated visual servoing for precision manipulation. In *International Conference on Robotics and Automation (ICRA)*, 1997.

[12] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[13] J. Kober, E. Oztop, and J. Peters. Reinforcement learning to adjust robot movements to new situations. In *Robotics: Science and Systems (RSS)*, 2010.

[14] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *International Journal of Robotic Research*, 32(11):1238–1274, 2013.

[15] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *International Conference on Robotics and Automation (IROS)*, 2004.

[16] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. 2012.

[17] T. Lampe and M. Riedmiller. Acquiring visual servoing reaching and grasping skills using neural reinforcement learning. In *International Joint Conference on Neural Networks (IJCNN)*, 2013.

[18] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *International Conference on Machine Learning (ICML)*, 2009.

[19] S. Levine and P. Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[20] S. Levine and V. Koltun. Guided policy search. In *International Conference on Machine Learning (ICML)*, 2013.

[21] S. Levine and V. Koltun. Variational policy search via trajectory optimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2013.

[22] S. Levine and V. Koltun. Learning complex neural network policies with trajectory optimization. In *International Conference on Machine Learning (ICML)*, 2014.

[23] S. Levine, N. Wagener, and P. Abbeel. Learning contact-rich manipulation skills with guided policy search. In *International Conference on Robotics and Automation (ICRA)*, 2015.

[24] W. Li and E. Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pages 222–229, 2004.

[25] W. Meeussen, M. Wise, S. Glaser, S. Chitta, C. McGann, P. Mihelich, E. Marder-Eppstein, M. Muja, Victor Eruhimov, T. Foote, J. Hsu, R.B. Rusu, B. Marthi, G. Bradski, K. Konolige, B. Gerkey, and E. Berger. Autonomous door opening and plugging in with a personal robot. In *International Conference on Robotics and Automation (ICRA)*, 2010.

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with deep reinforcement learning. *NIPS '13 Workshop on Deep Learning*, 2013.

[27] K. Mohta, V. Kumar, and K. Daniilidis. Vision based control of a quadrotor for perching on planes and lines. In *International Conference on Robotics and Automation*

(ICRA), 2014.

[28] I. Mordatch and E. Todorov. Combining the benefits of function approximation and trajectory optimization. In *Robotics: Science and Systems (RSS)*, 2014.

[29] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *International Conference on Robotics and Automation (ICRA)*, 2009.

[30] B. Pepik, M. Stark, P. Gehler, and B. Schiele. Teaching 3D geometry to deformable part models. In *Computer Vision and Pattern Recognition (CVPR)*, 2012.

[31] J. Peters and S. Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008.

[32] J. Peters, K. Mülling, and Y. Altün. Relative entropy policy search. In *AAAI Conference on Artificial Intelligence*, 2010.

[33] D. Pomerleau. ALVINN: an autonomous land vehicle in a neural network. In *Advances in Neural Information Processing Systems (NIPS)*, 1989.

[34] M. Riedmiller, S. Lange, and A. Voigtlaender. Autonomous reinforcement learning on raw visual input data in a real world application. In *International Joint Conference on Neural Networks*, 2012.

[35] S. Ross, G. Gordon, and A. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *Journal of Machine Learning Research*, 15:627–635, 2011.

[36] S. Savarese and L. Fei-Fei. 3D generic object categorization, localization and pose estimation. In *International Conference on Computer Vision (ICCV)*, 2007.

[37] P. Y. Simard, D. Steinkraus, and J. C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Seventh International Conference on Document Analysis and Recognition*, 2003.

[38] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[39] R. Tedrake, T. Zhang, and H. Seung. Stochastic policy gradient reinforcement learning on a simple 3d biped. In *International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[40] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in Neural Information Processing Systems (NIPS)*, 2014.

[41] H. Wang and A. Banerjee. Bregman alternating direction method of multipliers. In *Advances in Neural Information Processing Systems (NIPS)*. 2014.

[42] W. J. Wilson, C. W. Williams Hulls, and G. S. Bell. Relative end-effector control using cartesian position based visual servoing. *IEEE Transactions on Robotics and Automation*, 12(5), 1996.

[43] B. H. Yoshimi and P. K. Allen. Active, uncalibrated visual servoing. In *International Conference on Robotics and Automation (ICRA)*, 1994.

APPENDIX

A. Guided Policy Search Algorithm Details

In this appendix, we describe a number of implementation details of our BADMM-based guided policy search algorithm.

1) BADMM step size and weight adjustment: Recall that the inner loop alternating optimization is given by

$$\theta \leftarrow \arg\min_{\theta} \sum_{t=1}^{T} E_{p(\mathbf{x}_t)\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)}[\mathbf{u}_t^{\mathrm{T}}\lambda_{\mu t}] + \nu_t \phi_t^\theta(\theta, p)$$

$$p \leftarrow \arg\min_{p} \sum_{t=1}^{T} E_{p(\mathbf{x}_t,\mathbf{u}_t)}[\ell(\mathbf{x}_t,\mathbf{u}_t) - \mathbf{u}_t^{\mathrm{T}}\lambda_{\mu t}] + \nu_t \phi_t^p(p,\theta)$$

$$\lambda_{\mu t} \leftarrow \alpha\nu_t(E_{\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t)}[\mathbf{u}_t] - E_{p(\mathbf{u}_t|\mathbf{x}_t)p(\mathbf{x}_t)}[\mathbf{u}_t]).$$

We use a step size of $\alpha = 0.1$ in all of our experiments, which we found to be more stable than $\alpha = 1.0$. The weights ν_t are initialized to 0.01 and incremented based on the following schedule: at every iteration, we compute the average KL-divergence between $p(\mathbf{u}_t|\mathbf{x}_t)$ and $\pi_\theta(\mathbf{u}_t|\mathbf{x}_t)$ at each time step, as well as its standard deviation over time steps. The weights ν_t corresponding to time steps where the KL-divergence is higher than the average are increased by a factor of 2, and the weights corresponding to time steps where the KL-divergence is two standard deviations or more below the average are decreased by a factor of 2. The rationale behind this schedule is to adjust the KL-divergence penalty to keep the policy and trajectory in agreement by roughly the same amount at all time steps. Increasing ν_t too quickly can lead to the policy and trajectory becoming "locked" together, which makes it difficult for the trajectory to decrease its cost, while leaving it too low requires more iterations for convergence. We found this schedule to work well across all tasks, both during trajectory pre-training and while training the visuomotor policy.

2) Policy variance optimization: As discussed in the paper, the variance of the Gaussian policy $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ does not depend on the observation, though this dependence would be straightforward to add. Analyzing the objective $\mathcal{L}_\theta(\theta, p)$, we can write out only the terms that depend on Σ^π:

$$\mathcal{L}_\theta(\theta, p) = \frac{1}{2N} \sum_{i=1}^{N} \sum_{t=1}^{T} E_{p_i(\mathbf{x}_t,\mathbf{o}_t)} \left[\mathrm{tr}[\mathbf{C}_{ti}^{-1}\Sigma^\pi] - \log|\Sigma^\pi| \right].$$

Differentiating and setting the derivative to zero, we obtain the following equation for Σ^π:

$$\Sigma^\pi = \left[\frac{1}{NT} \sum_{i=1}^{N} \sum_{t=1}^{T} \mathbf{C}_{ti}^{-1} \right]^{-1},$$

where the expectation under $p_i(\mathbf{x}_t)$ is omitted, since $\mathbf{C}_{ti}$ does not depend on $\mathbf{x}_t$.

3) Trajectory optimization: In this section, we review how the LQR backward pass can be used to optimize the constrained objective in Section IV-B. This derivation follows previous work [19]. The constrained trajectory optimization problem is given by

$$\min_{p(\tau)\in\mathcal{N}(\tau)} \mathcal{L}_p(p,\theta) \text{ s.t. } D_{\mathrm{KL}}(p(\tau)\|\hat{p}(\tau)) \leq \epsilon.$$

As discussed in the paper, $\mathcal{L}_p(p,\theta)$ can be written as the expectation of some function $c(\tau)$ that is independent of p, such that $\mathcal{L}_p(p,\theta) = E_{p(\tau)}[c(\tau)]$. Writing the Lagrangian of the constrained optimization, we have

$$\mathcal{L}(p) = E_{p(\tau)}[c(\tau)] - \eta E_{p(\tau)}[\log\hat{p}(\tau)] - \eta\mathcal{H}(p(\tau)) - \eta\epsilon,$$

where η is the Lagrange multiplier. Note that $\mathcal{L}(p)$ is the Lagrangian of the constrained trajectory optimization, which is not related to the augmented Lagrangian $\mathcal{L}_p(\tau,\theta)$. Grouping the terms in the expectation and omitting constants, we can rewrite the minimization of the Lagrangian with respect to the primal variables as

$$\min_{p(\tau)\in\mathcal{N}(\tau)} E_{p(\tau)} \left[\frac{1}{\eta} c(\tau) - \log\hat{p}(\tau) \right] - \mathcal{H}(p(\tau)).$$

Let $\tilde{c}(\tau) = \frac{1}{\eta}c(\tau) - \log\hat{p}(\tau)$. The above optimization then corresponds to minimizing $E_{p(\tau)}[\tilde{c}(\tau)] - \mathcal{H}(p(\tau))$. As shown in prior work (see, e.g., [20]), this type of maximum entropy problem can be solved using the LQR algorithm, and the solution is given by

$$p(\mathbf{u}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t; Q_{\mathbf{u},\mathbf{u}t}^{-1}),$$

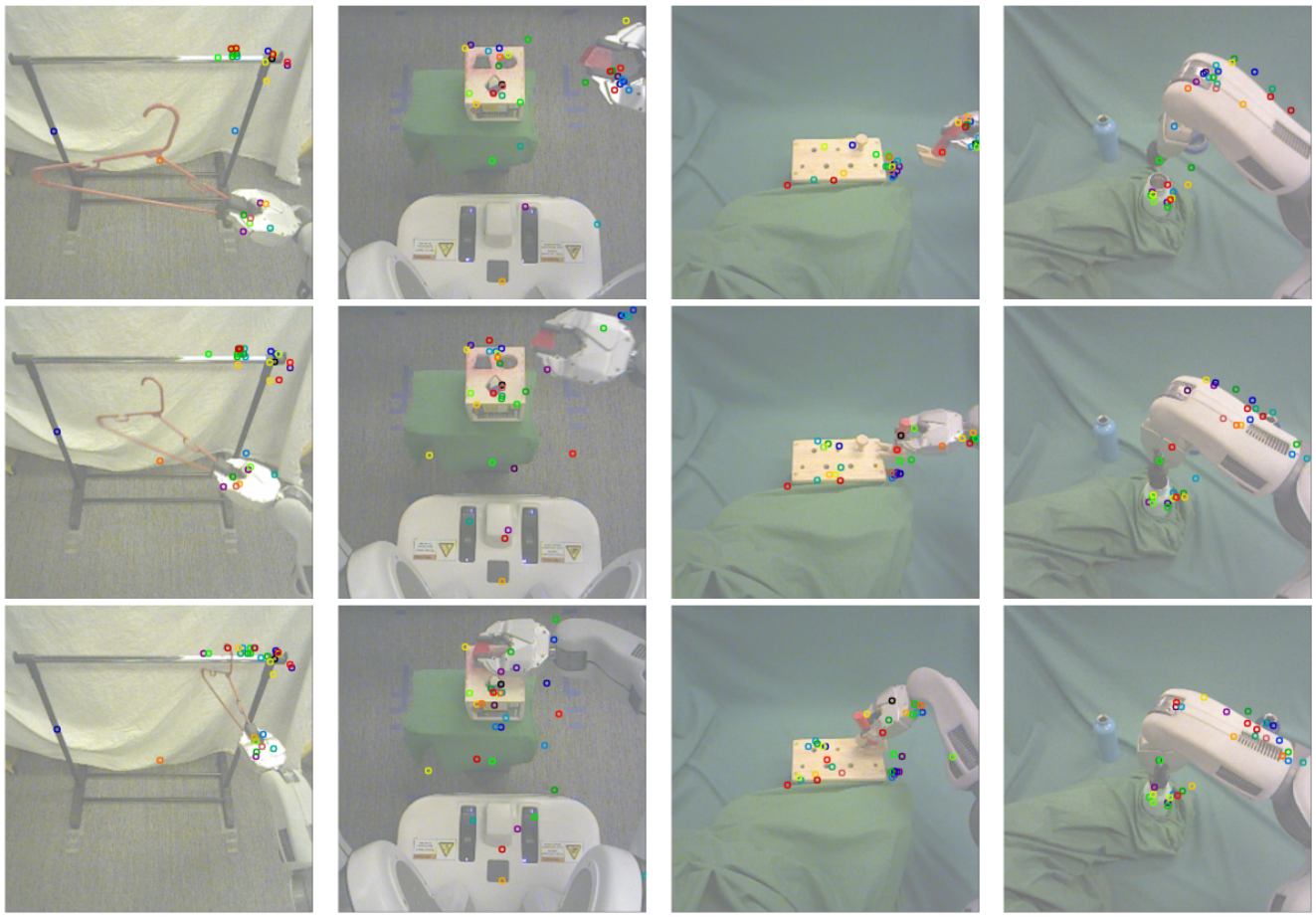
where $\mathbf{K}_t$ and $\mathbf{k}_t$ are the feedback and open loop terms of the optimal linear feedback controller corresponding to the cost $\tilde{c}(\mathbf{x}_t,\mathbf{u}_t)$ and the dynamics $p(\mathbf{x}_{t+1}|\mathbf{x}_t,\mathbf{u}_t)$, and $Q_{\mathbf{u},\mathbf{u}t}$ is the quadratic term in the Q-function at time step t. All of these terms are obtained as a result of the standard LQR backward pass (see, e.g., [24]).

B. Feature Point Analysis

The visual processing layers of our architecture automatically learn features points using the fixed transformation from the softmax to spatial coordinates. These feature points encapsulate all of the visual information received by the motor layers of the policy. In Figure 6, we show the features points discovered by our visuomotor policy through guided policy search. Each policy learns features on the target object and the robot manipulator, both clearly relevant to task execution. The policy tends to pick out robust, distinctive features on the objects, such as the left pole of the clothes rack, the left corners of the shape-sorting cube, the bottom-left corner of the toy tool bench, and the edges of the bottle.

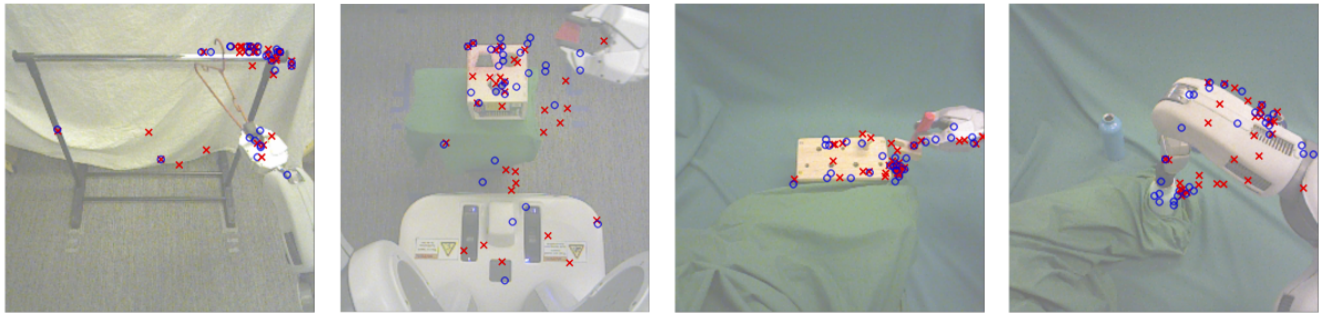
In Figure 7, we compare the feature points learned through guided policy search to those learned by a CNN trained for pose prediction. Note that in all tasks, the end-to-end trained model produces fewer points on the background compared to the model trained on object pose. In the bottle task, the end-to-end trained policy outputs points on both sides of the bottle, including one on the cap, while the pose prediction network only finds points on the right edge of the bottle.

The feature point representation is very simple, since it assumes that the learned features are present at all times. While this is a drastic simplification, both the pose predictor and the policy still achieve good results. A more flexible architecture that still learns a concise feature point representation could further improve policy performance. We hope to explore this in future work.



(a) hanger (b) cube (c) hammer (d) bottle

Fig. 6: Feature points tracked by the policy during task execution for each of the four tasks. Each feature point is displayed in a different random color, with consistent coloring across images. The policy finds features on the target object and the robot gripper and arm. In the bottle cap task, note that the policy correctly ignores the distractor bottle in the background, even though it was not present during training.



(a) hanger (b) cube (c) hammer (d) bottle

Fig. 7: Feature points learned for each task. For each input image, the feature points produced by the policy are shown in blue, while the feature points of the pose prediction network are shown in red. The end-to-end trained policy tends to discover more feature points on the target object and the robot arm than the pose prediction network.